

Technical Field

[0001]

The present invention relates to a software generation
5 method based on a mathematical way of thinking, and more
particularly to a software generation method which uses an
adjacency matrix and topological sort for Lyee methodology.

Background Art

[0002]

10 (1) Problems of software development

Easy and quick manufacturing of high-quality software is
a basic concern in a software development and research field.
Various methodologies and technologies have been invented and
proposed to improve productivity, maintenance efficiency, and
15 quality of software developments. However, none have been
able to achieve such tasks. One of the reasons is that
software itself is complex and difficult to be understood.
Another reason is a limit on a current methodology. In fact,
almost all methodologies that have been proposed have not
20 failed to manufacture clearly understood and modifiable
systems, and they are still considered to be for use by
specialists who have a very wide range of skills and technical
knowledge and know-how. Thus, labor and maintenance costs
become high, and extensive checks are necessary to be executed
25 on software. Outlines and problems of conventional software
development method are summarized as follows.

(2) Procedural programming

Generally, procedural programming has been effective in
development and implementation of business systems because of

easy handling of its descriptive language. However, important problems still remain to be solved, such as a difficulty of accurate implementation of requirements, productivity, and maintenance.

5 (3) Declarative programming

 Meanwhile, declarative programming, a programming of making declarations which is requirement definition itself, has been used in an application field of a noncalculation type such as logical reasoning, inference and user intelligence gathering of artificial intelligence or an expert system. The declarative programming is a method not to describe control procedure, but to declare relations between data defined in requirements. As its programming is executed based on the declaration, an advantage of the declarative programming is that the requirements can be accurately implemented while the procedural programming requires to explicitly specify procedure such as control procedure.

[0003]

 However, even in the declarative programming, there is a problem that performance of software in running is not good because of execution control structures. There is also a disadvantage that it is difficult to learn a good command of languages for the declarative programming. As languages for the declarative programming, there are a logical programming language (e.g., PROLOG) which uses a logical equation, and a functional programming language (e.g., LISP) which describes a target state as a mathematical function. No value is generated only by a declaration. To generate a value, therefore, a declarative language has a procedural logical

mechanism to execute the declaration. Such a procedural logical mechanism has many conditions to enable execution of all types of declarations, which requires knowledge of mathematics and logic, and thus its use is difficult.

5 Accordingly, the declarative language has not been in widespread use. This is why the procedural programming language by a procedural language which has a structure close to that of a natural human language has been in widespread.

(4) Object oriented programming

10 We have object oriented programming, which is a programming technique for integrating data and an operation procedure (called method) into a module called a object, and describing a program as a combination of objects. The program comprises highly independent modules called objects.
15 Thus, this programming provides advantages that range of influences caused by changes/modifications is limited, easy re-use of each object, and the like.

[0004]

20 However, there are no strict rules on what range has to be one object to implement requirements, and how components are combined, and no regulations on an execution order such as a structured design approach. In an actual development field, relations between objects are not applied with complete consistency. In many cases, even if great many objects are
25 produced as a result of developments, nobody except creators themselves understands the objects. In other words, completed software becomes a group of functions of no regularities, and has a strong tendency that it is finally difficult to decompose and reuse them.

[Patent Document 1]

Pamphlet of International Publication No. 97-16784

[Patent Document 2]

Pamphlet of International Publication No. 89-19232

5 [Patent Document 3]

Pamphlet of International Publication No. 99-49387

[Patent Document 4]

Pamphlet of International Publication No. 00-79385

[Patent Document 5]

10 Pamphlet of International Publication No. 02-42904

[Patent Document 6]

Pamphlet of International Publication No. 2004-68342

[Patent Document 7]

Publication of Japanese Patent Application Laid-Open No.

15 2002-202883

[Nonpatent Document 1]

"The Formal Semantics of Programming Languages" by Glynn Winskel, the MIT Press, 1993

[Nonpatent Document 2]

20 "Semantics with Applications; A Formal Introduction" by Hanne Riis Nielson, and Fleming Nielson, John Wiley & Sons, 1992

[Nonpatent Document 3]

25 "Principle of Lyee Software" by Fumio Negoro, pp. 441 to 446, Proceedings of 2000 International Conference on Information Society in the 21st Century (IS2000), Aizu, Japan, Nov. 5 to 8, 2000

[Nonpatent Document 4]

"Intent Operationalisation for Source Code Generation"

by Fumio Negoro, The 5th World Multi-Conference on Systemics,
Cybernetics and Informatics (SCI 2001), Orland, US, July 22 to
25, 2001

[Disclosure of the Invention]

5 [Problems to be solved by the Invention]

[0005]

(1) Lyee development methodology

Recently, there has been proposed a new and very
promising methodology called Lyee (tail-character word of
10 "governmental methodology for software providence", read as
"lee", and invented by Fumio Negoro). The Lyee is a new
method of automatically developing software from requirements.

[0006]

In the case of the Lyee methodology, while it is a
15 declarative type, no special logical mechanism is necessary to
execute declarations, and the execution of the declarations is
a procedure of simple value substitution as in the case of the
procedural programming. Accordingly, a language to be used
can be generally used procedural languages. In other words,
20 the Lyee methodology has advantages of both of a declarative
type which enables accurate implementation of requirements and
a procedural type which can be written by procedural languages
of easy description.

[0007]

25 For an outline of the Lyee development method, Patent
Application No. 2004-27240 by the applicant of the present
invention, and the Patent Document 6 will be quoted, and
regarded as parts of the disclosure. The outline is as
follows.

(2) Declaration in Lyee methodology

According to the Lyee methodology, an abstraction target to be declared is a variable. A declared variable is called a word according to the Lyee methodology. Generally,

5 declarative specifications mean that abstracted unit to be declared can written in any order, and the units are described using an equation, and there is bi-directionality from a right side to a left side, and from the left side to the right side. According to the Lyee methodology, however, there is no bi-
10 directionality while descriptive ordinality of words is removed. Thus, a framework is simple.

[0008]

Main components of a word declaration are a word name, a definition equation, calculation conditions of the definition
15 equation, input/output attributes, value attributes, and the like. The definition equation is a relation equation showing a relation with other words. For example, a word a can be declared by a definition equation $a = b + c$. For execution of this declaration, the relational equation only needs to be
20 executed as a substitution equation. For example, to execute the declaration of word a, $a = b + c$, (to obtain a value of a), it is only needed to substitute values for the variables b and c on the right side of a.

(3) Mechanism for executing declaration within framework
25 of procedural type

It is a fixed program structure provided by the Lyee methodology that realizes orderless description of declarations while it is implemented in a framework of procedural languages. As long as a computer for processing a

program is a currently used von Neumann type which sequentially processes commands, in the case of the procedural language, a problem of order must be solved in some way. In the case of the other declarative programming methods, it is solved by declarative language. According to the Lyee methodology that uses the procedural language, the problem of order is solved by the program structure.

[0009]

The program structure comprises modules of fixed structures for executing declarations of word units (called "declaration execution modules" hereinafter). These modules are grouped into a unit called a palette. In the palette, any order can be employed to execute the declaration execution modules. An execution control module controls execution of the declaration execution modules to be repeated until execution of all the execution modules in the palette is completed (i.e., until values are generated).

[0010]

The palettes are a W02 palette for executing declarations of input words, a W03 palette for executing calculation conditions in declarations of output words, and a W04 palette for executing definition equations and outputs in the declarations of the output words. A set of these three kinds of palettes constitutes a collective unit called a basic structure. That is, a program of Lyee structure employs a structure that it is divided into basic structure units, the basic structure is divided into three kinds of palettes, and each palette is divided into declaration execution modules. The basic structure is made for each output logical record

(i.e., a group of words simultaneously output to the same medium). A process route diagram (PRD) shows this division of structure of the Lyee program.

[0011]

5 The entire program of the Lyee methodology comprises the declaration execution modules of fixed structures for executing the declaration of word units. Besides, a method of constituting the entire program of those declaration execution modules has been fixed. Accordingly, a program can be
10 automatically generated by substituting requirements of word units for the declaration execution modules of the fixed structures.

[0012]

 In reality, automatic generation of a program by the
15 Lyee methodology is realized by a tool called "LyeeALL". The LyeeALL is a tool for constructing one program by automatically substituting a declaration of requirements in given blank sections of codes of the declaration execution modules of the fixed structures (called "template" meaning a
20 form), and constituting the declaration execution modules according to the given rule.

[0013]

 In software developments based on the Lyee methodology, engineers are divided into two groups, and respectively carry
25 out two kinds of specialized work. One team is skilled in an external environment, such as OS, middleware or database, in which the program of the Lyee structure is executed, and they set parameters of each external environment in the templates of the Lyee structure program. The other team defines user

requirements, and declares the requirements according to the Lyee methodology to implement the requirements in the templates of the Lyee structure. As developments based on such a division system of labor are available, in the developments based on the Lyee methodology, all programmers do not need to have a wide range of skills concerning the external environments, and a problem of a shortage of workers with such skills can be solved.

(4) Problems of conventional Lyee structure

As described above, the Lyee methodology has great advantages of accurate implementation of the requirements, automatic code generation, easy maintenance, and the like. However, there are problems in the iteration and divided structures. It is said that an execution speed is low because of the iteration processing and that a program size is large because of the divided structures.

[0014]

On the other hand, according to the Lyee methodology, the requirements are divided into units called basic structures to implement specifications. Explicit information regarding execution orders such as screen transitions, key relations of database or the like are captured by a group of words (record), and an execution condition of the words belonging to the group are collectively recognized. In the current Lyee structure, since these divided structures are directly formed into codes, the program size is large, and the execution speed is low.

[0015]

It is an important task to improve the Lyee structure

into a program of no unnecessary iteration, a smaller size, and an efficient processing speed while maintaining its declarative programming advantages and requirement meanings without any changes.

5 [0016]

As a method of improving the execution speed by eliminating the unnecessary iteration, there has been proposed a method of automatically optimizing an order of declaration execution modules (ordering to complete declaration execution
10 by one round) by topological sort (PCT/JP 03/09591). By applying topological sort to modules in a palette, it is possible to avoid useless iteration in the palette.

[0017]

However, there is a limit on effects if only the
15 topological sort is applied while the framework of the Lyee divided structures is maintained. It is because there are also iterations between the palettes and between the basic structures, and generation of all values is not completed by one round even while the number of iteration times is reduced
20 to a minimum by the optimal ordering in the palette by the topological sort. Additionally, the divided structures themselves cause an increase in program size.

[0018]

Thus, to bring greater solutions to the problems of the
25 size and the processing speed of the Lyee program having the iteration and divided structures, it may be effective not to divide into units called the palettes and the basic structures and to apply topological sort to declaration execution modules in a range of requirements implemented as one program.

[0019]

The present invention relates to a method of applying topological sort to an entire program by eliminating the divided structures of the conventional Lyee structure. Its object is, therefore, to improve a software execution speed and a program size based on the Lyee methodology.

[Means for Solving the Problem]

[0020]

To achieve the object of the present invention, a software generation method comprises: a first step for defining a statement execution unit of any of L2 processing (checking process for input word's attribute), L processing (value generation processing of output word), I2 processing (logical body input processing), and O4 processing (logical body output processing), all of which are necessary for satisfying the requirements, from word-unit statements in which the user requirements to be implemented as a program is declared by a word name, a definition equation, execution conditions of the definition equation, input/output attributes, and attributes of a word value for each logical body accompanied by access conditions and for each word on the logical body; a second step for defining a (partial) order relation of all the defined L2 processing (checking process for input word's attribute), L processing (value generation processing of output word), I2 processing (logical body input processing), and O4 processing (logical body output processing); a third step for executing topological sort for the L2 processing, L processing, I2 processing, and O4 processing defined in the (partial) order relation defined in

the second step; and a fourth step for arranging a predetermined code sequence based on Lyee methodology and relevant to the statement execution unit in accordance with an order of the statement execution units rearranged in the third
5 step.

[0021]

According to the present invention thus constructed, a program is created to execute the statement of word units defined in the first step by units of the L2 processing
10 (checking process for input word's attribute), the L processing (value generation processing of output word), the I2 processing (logical body input processing), and the O4 processing (logical body output processing) in a shortest order, whereby software generation using the Lyee methodology
15 becomes more efficient. In other words, since all the functions of the steps can be automated, it is possible after all to automate a process consistently from elimination of iteration processing to software generation. Thus, a software production speed and efficiency can be greatly improved.

20 [0022]

Here, "definition of (partial) order relation among all the L2 processing (checking process for input word's attribute), L processing (value generation processing of output word), I2 processing (logical body input processing),
25 and O4 processing (logical body output processing)" means that the order relation among them is represented by, for example, a directed graph by focusing on the words used in the statement. As the order relation, a partial order relation and total order relation are conceivable. The "directed

graph" represents, by a node, each statement implementation unit contained in requirements from a user who requests a software development, and a partial order relation (or total order relation) $a \leq b$ (" \leq " is defined as a symbol to mean that b is created from a, similar hereinafter) between start and end points by an arrow. For example, software having such a function programmed therein, or a read-only memory (ROM) having the coded software written therein may be used.

[0023]

The "topological sort" is a function of executing depth-first search (described later) against the obtained directed graph, and of aligning node columns (sequences of numbers) while obtaining nodes during return through a search path from a reached end. For example, software having such a function programmed therein, or a read-only memory (ROM) having the coded software written therein may be used. This topological sort may be operated to generate a nilpotent matrix by representing the obtained directed graph by an adjacency matrix and executing depth-first search against the directed graph.

[0024]

Here, the "depth-first search" is an algorithm of:

(1) advancing through interconnected nodes starting from a start-point node to avoid overlapping,

(2) returning to a node having a place to visit when there are no more places to visit (comeback),

(3) advancing again from the node to which the process has returned as in the case of (1), and

(4) finishing the process when there are more places to

advance to.

[0025]

Here, the "search" means various algorithms including, in addition to the aforementioned depth-first search, breadth-
5 first search, iterative deeping, heuristic search, hill climbing, best-first search, Euler circuit, Dijkstra method, and the like. For example, software having such functions programmed therein, or a read-only memory (ROM) having the coded software written therein may be used.

10 [0026]

The "Software" used here has a broad meaning. That is, it is a concept including both of a development request from a user and an object program which satisfies the request.

[0027]

15 With the foregoing configuration, according to the present invention, since the iteration is prevented, it is possible to develop software of higher efficiency, a higher speed, and higher performance. Moreover, preprocessing for eliminating iteration is automatically executed, and an object
20 program is generated by the Lyee (registered trademark) methodology based on the obtained preprocessed program of a statement execution unit. In other words, it is possible to automate a process from sophistication (alignment) of user requirements to generation of the object program. Thus, the
25 invention provides great effects to the software industry, such as great increases in efficiency of software production, productivity, and quality.

[0028]

According to a different embodiment of the present

invention, each of a program (software) for generating
"development target software", a program generator, a program
processor, a tool (including both as a device and as a
software), a software development device, a software
5 development support device, and a software development
management device can be configured by comprising: means for
defining a statement execution unit of any of L2 processing
(checking process for input word's attribute), L processing
(value generation processing of output word), I2 processing
10 (logical body input processing), and O4 processing (logical
body output processing), all of which are necessary for
satisfying the requirements, from word-unit statements in
which the user requirements to be implemented as a program is
declared by a word name, a definition equation, execution
15 conditions of the definition equation, input/output attributes,
and attributes of a word value for each logical body
accompanied by access conditions and for each word on the
logical body; means for defining a (partial) order relation of
all the defined L2 processing (checking process for input
20 word's attribute), L processing (value generation processing
of output word), I2 processing (logical body input processing),
and O4 processing (logical body output processing); means for
executing topological sort for the L2 processing, L processing,
I2 processing, and O4 processing defined in the (partial)
25 order relation defined in the second step; and means for
arranging a predetermined code sequence based on Lyee
methodology and relevant to the statement execution unit in
accordance with an order of the statement execution units
rearranged in the third step.

[0029]

Furthermore, the present invention is realized by the software produced by the aforementioned "method of generating development target software", and a recording medium having the software mounted thereon, or a device (hardware) having the software mounted thereon. In this case, the present invention can be constructed as codes in which statement execution units of all necessary for satisfying the requirements, L2 processing (attribute check processing of input word), L processing (value generation processing of output word), I2 processing (logical body input processing), and O4 processing (logical body output processing), all of which are defined by word-unit statement declared based on a word name, a definition equation, execution conditions of the definition equation, input/output attributes, and attributes of a word value, the word-unit statement stating the user requirement to be implemented as a program for each logical body accompanied by access conditions and for each word on the logical body, can be configured as a code sequence based on the Lyee methodology in accordance with an order rearranged by topological sort executed based on a (partial) order relation defined from the requirements declared by the word units.

[0030]

The present invention is realized as software as a software code proformas used for producing the software by the aforementioned "method of generating development target software", and a recording medium having the software mounted thereon or a device (hardware) having the software mounted thereon.

[0031]

Furthermore, the present invention can be realized as an extraction method of information (document (paper, data)) extracted from the requirements by the aforementioned "method of generating development target software), the information (document (paper, data)) extracted by the extraction method, a method of using the extracted information, an information recording medium having such information mounted therein, software having the coded information extraction method/using method, a recording medium/device (hardware) having the software mounted thereon, or information extracted from software development requirements having correlated pieces of information to enable realization of such.

[0032]

For the logical record, Japanese Patent Application No. 2004-272400 by the same applicant will be referred to, quoted, and made a part of the disclosure.

[Advantages of the Invention]

[0033]

According to the present invention, in the program comprising the word unit declaration execution modules, the declaration execution module processing for generating output data can be completed within the minimum number of execution times by avoiding useless iteration, and the program size can be reduced. Specifically, the following points are enabled.

[0034]

The declaration execution units of word units to be executed can be decided from the declaration of word units declaring the program requirements, and a partial order

relation can be defined among the declaration execution units of the word units.

[0035]

Some of declaration-by-the-word execution units become
5 unnecessary by the optimal ordering, for example, the routing
action element, and they are removed. For execution
conditions of the routing action element which is information
of order to be implemented to maintain requirement meanings,
the requirement meanings can be maintained by reflecting the
10 execution conditions in partial order relation definition of
the other declaration-by-the-word execution units.

[0036]

The topological sort is applied to the group of the
declaration-by-the-word execution units (not including the
15 routing action element) to rearrange them in an optimal order.
Thus, the program can be executed, avoiding unnecessary
iteration, in a minimum number of execution times.

[Best Modes for Carrying Out the Invention]

[0037]

20 The present invention is designed to solve the foregoing
problems by eliminating the divided units of the conventional
Lyee structure program and applying topological sort to the
entire program. By the invention, an improvement in execution
speed of software developed by the Lyee methodology and an
25 improvement in program size are realized. Hereinafter,
referring to the drawings, specific embodiments of the present
invention will be described. For a technology of applying the
topological sort for the Lyee structure program, the Patent
Document 6 by the same applicant will be quoted and made a

part of the disclosure.

[0038]

Chapter 1 gives an outline of divided structures of Lyee software which are integration targets of the present invention. Chapter 2 describes a specific method of integrating basic structures. Chapter 3 describes an example of a program with integrated basic structures by using the present invention and its effects.

<Chapter 1: mechanism of divided structures of Lyee software>

To integrate the divided structures of the conventional Lyee structure program, first, a mechanism of the divided structures must be understood. Dividing into basic structures is to declare, based on requirements, how words should be grouped to output and with what conditions they should be output. Declaration elements of word units include input/output attributes. In addition to its input/output classification, requirements to be declared are input or output groups (logical record) to which each word belongs and conditions under which the input/output groups are input or output. For details on how requirement information regarding this input/output is accurately taken out from a requirement definition indicating various processing operations, Japanese Patent Application No. 2004-272400 by the same applicant will be quoted, and made a part of the disclosure. Here, regarding a meaning of the division into the basic structures, an outline only considered necessary for explanation of the present invention will be given.

1. Meaning of division into basic structures

Meaning of divided structures

The divided structures of the Lyee software are made by grouping declaration execution modules which are modules of minimum units in accordance with a certain rule. The structures are shown in FIG. 1.

5 1) Synchronization range

Modules constituting a program are first grouped into a unit indicated by "synchronization range" in (a) of FIG. 1. The synchronization range is a range of processes which are defined in requirement to be executed by a computer by an event that is a user instruction to the program of executing processes (e.g., command of pressing a button, selecting a menu, or the like). In other words, the synchronization range is a group of modules to be executed by the same event as an execution condition. According to the Lyee methodology, completion of processes in the synchronization range is called "synchronization". The completion of the processes in the synchronization range means that generation and outputting of output words defined by the requirements to be executed by the event are all completed.

20 2) Basic structure

Next, as indicated by 101 in (a) of FIG. 1, modules within the synchronization range are grouped into basic structures. Input/output processing of the computer is executed not by the data but by the set of data. A set of words simultaneously input to/output from the same definition body is called a logical record according to the Lyee methodology. The logical record is a unit to hold a value of a word such as a screen, a file, a form, or a transmission. It is a basic structure that groups modules regarding the same

output logical record. A role of the program is to generate and output a value of an output word specified by user's event. However, the number of outputs made within the synchronization range by one event is not limited to one. In many cases, two or more output logical record are output within one synchronization range, e.g., saving (outputting) of a generation result of word values in a file in addition to displaying (outputting) of them on a screen, and synchronization ranges are grouped into a plurality of basic groups.

3) Palette

Lastly, modules in the basic structure are grouped into three kinds. A group of modules in the basic structure is called a palette according to the Lyee methodology. Three palettes are called a W02 palette, a W03 palette, and a W04 palette. The W02 palette is a set of modules regarding inputting of input words. The W03 palette is a set of modules regarding determination of calculation conditions of definition equations of output words. The W04 is a set of modules regarding generation of values by execution of the definition equations of the output words, and outputting of the values.

[0039]

Next, an execution order of the basic structures will be described.

1) Within same synchronization range

There is a start-point and end-point relation among the basic structures. Thus, a basic structure to be executed next is decided by depth-first search. An execution order is

decided by applying topological sort to the declaration execution modules within the same synchronization range.

2) Execution order of synchronization ranges

Which synchronization range should be executed depends
5 on an event given by the user. For example, there are buttons
for executing processing A and processing B. The user presses
one of these buttons to decide processes (i.e.,
synchronization range) to be executed by the program. Thus,
as the entire program comprises a plurality of synchronization
10 ranges having different events set as execution conditions,
topological sort cannot simply be applied to the declaration
execution modules of the entire program. In the Chapter 2, by
using specifications having two synchronization ranges of two
events, a method of integrally applying topological sort to
15 declaration execution modules within different synchronization
ranges will be described.

2. Target of topological sort

A target of optimal ordering by the topological sort is
processes executed by declaration execution modules which are
20 components of the Lyee software. There are two kinds of
declaration execution modules. One is called a logical
element that is a module to establish a word value. The other
is called an action element that is a module to execute
processes such as input/output other than that of value
25 establishment.

[0040]

The Patent Document 6 by the same applicant discloses an
embodiment of topological sort in which a logical element for
establishing a value is treated as an element of a directed

graph and an adjacency matrix and represented by a mathematical model. As in the case of the logical element, an action element can be treated as an element of a directed graph and an adjacency matrix, and can be a target of topological sort. A large difference between the logical element and the action element is only that the logical element act (action to generate a value) on only one word while the action element simultaneously act on two or more words. For example, a routing action element (specifying a basic structure to be executed next) act on a set of words called basic structures, and an output act element (outputting output words) act on a set of words called output logical record.

[0041]

Hereinafter, discussing on declaration execution modules one by one, a method of getting them a target of topological sort will be described.

[Logical element]

[L2]

L2 is a declaration execution module for checking attributes of input words with requirements. Under any conditions, conditions for enabling execution of the L2 processing are that a value of the input word has been fetched (input) from a medium into a memory area. Thus, the L2 process the topological sort be applied to is considered that its start point is I2 inputting a value of the L2.

[0042]

An event command (button or menu) is also treated as one of the input words.

[0043]

[L3 and L4]

In the current Lyee software structure, generation
process of an output word is realized by two modules of L3
5 (execution of calculation condition equation) and L4
(execution of definition equation). Further, when an output
word a has plural definition equations (word in this case is
called an equivalent word), L3 and L4 are provided by numbers
equal to the numbers of sets of definition equations and
10 calculation equations.

[0044]

It is assumed that the output word a has plural
definition equations (word in this case is called an
equivalent word) and declared as follows.

15 [0045]

<Definition equation>	<Calculation condition>
(1) $b + c$	$e > 10$
(2) $b - d$	$e \leq 10$

While the equivalent word has plural definition
20 equations, one value must always be given to the output word.
Thus, the calculation conditions for the definition equation
of the equivalent word must be exclusive and complete.

[0046]

Under any conditions, conditions for executing the
25 generation process of the word a is that all start-point words
used in all the definition equations and calculation condition
equations of the word a have been decided. In other words, as
for this target of topological sort, definition equations of
words b, c, d and e only need to be executed first.

[0047]

Accordingly, targets of topological sort are all sets of L3 and L4 processes for each output word, and their start points are L2 processes of an input word and a set of L4 and L3 processes of an output word, which are used as start points in the definition equations and the calculation conditions.

[Action element]

[Input action element I2]

I2 is a module to execute process of fetching (inputting) a value from a medium into a memory. Thus, under any conditions, conditions for execution of the I2 process are unconditional (i.e., nondependent on other processes). The I2 process as a target of topological sort does not have any start point (however, in the case of an input from a file, a value of a key word must be established, which is a key for accessing the file).

[Output action element O4]

O4 is a module for executing process of writing (outputting) values of words from a Lyee software area into an output buffer of the memory. Accordingly, under any conditions, conditions to execute the O4 process are that values of output words to be output has been established (i.e., one L4 has been executed). The O4 process as a target of topological sort can be regarded that its start points are output words to be output.

[0048]

[Structure action element S4]

S4 is a module to execute process of initializing (recording initial values) an area of recording a word value

and an area of recording a process result (e.g., output has been normally completed, or not completed). In the conventional Lyee having the iteration structure, prohibition of overwriting and initialization in the word value area are means for the mechanism of the iteration structure. Thus, in the Lyee structure optimally ordered by applying the topological sort to the entire program, the initialization of the word value areas becomes unnecessary.

[0049]

On the other hand, the initialization of the recording area of the processing result is also necessary for the integrated Lyee structure because it is not for the mechanism of the iteration structure.

[0050]

[Routing action element]

A routing action element is a module to execute process of specifying a palette to be executed next. The routing action element plays a role of linking palettes which are divided units of the program, and basic structures which constitute a set of palettes.

[0051]

<Routing action element for transition between palettes within a basic structure>

<Routing action element for transition between basic structures>

Hereinafter, a meaning of a routing action element for selecting a basic structure to be executed from one to another will be described by taking an example.

(Example 1)

FIG. 2 shows a screen of a system, where a user obtains a value of a data item g by inputting data to data items c and d and pressing an execution button a or b. A reference numeral 701 denotes an input data field of the data item c, a reference numeral 702 denotes an input data field of the data item d, a reference numeral 703 denotes an output data field of the data item g, a reference numeral 704 denotes the execution button a, and a reference numeral 705 denotes the execution button b. When the execution button a is pressed, a value of g is calculated by a definition equation $c + d$. When the execution button b is pressed, a value of g is calculated by a definition equation $c \times d$. As they are never pressed simultaneously, the execution buttons a and b are conditions for executing one of the two definition equations to generate the value of g.

[0052]

A program of the screen of FIG. 2 is as shown in FIG. 3 when it is represented by divided units of basic structures. BS1, BS2, and BS3 of FIG. 3 are all basic structures. The basic structure BS1 is a basic structure of an output logical record to the screen shown in FIG. 2. The BS1 includes input words a and b which are execution buttons, input words c and d which are input data items, an output word g which is an output data item, and a routing action element X_r .

[0053]

The basic structure BS2 is a basic structure (medium is a file) for calculating a value of the output data item g by the equation $c + d$ when the equation button a is pressed. It includes a word e to record a result of the equation $c + d$.

The basic structure BS3 is a basic structure (medium is a file) for calculating a value of the output data item g when the execution button b is pressed. It includes a word f to record a calculation result of the equation $c \times d$.

5 [0054]

The routing action element X_r selects execution of one of the basic structures BS2 and BS3 for calculating the value of the output word g after the user inputs data to the screen (BS1 is executed). Words belonging to the two basic structures are established depending on which of the buttons a and b are pressed. Accordingly, the routing action element X_r selects the BS 2 as a basic structure to be executed next when the execution button a is pressed (word a = true, there is a value), and the BS3 as a basic structure to be executed next when the execution button b is pressed (word b = true, there is a value). Execution moves to a next basic structure after input data have been prepared. Thus, it is also a execution condition of the routing action element X_r that values are in the words c and d.

20 [0055]

By executing the basic structure BS2 or BS3, a value of the output word g is established in the word e or f. The output word g of the basic structure BS1 for outputting the output data to the screen is defined as $g = e$ when a value is established in the word e ($e = \text{true}$), and as $g = f$ when a value is established in the word f ($f = \text{true}$).

[0056]

The aforementioned definitions of the words are shown in Table 1.

[0057]

[Table 1]

Word	Input/output attribute	Definition equation	Definition equation execution conditions
a	Input	-	-
b	Input	-	-
c	Input	-	-
d	Input	-	-
x_r	(Routing action element)	BS2 is selected	(1) $a = \text{true}$ (button a is pressed, and a has a value), $c = \text{true}$, and $d = \text{true}$
		BS3 is selected	(2) $b = \text{true}$ (button b is pressed, and b has a value), $c = \text{true}$, and $d = \text{true}$
e	Output	$c + d$	x_r selects BS2
f	Output	$d \times d$	x_r selects BS3
g	Output	e	$e = \text{true}$ (e has a value)
		f	$f = \text{true}$ (f has a value)

[Control module]

5 In the conventional Lyee structure, there are divided structures, and iterations occur in each divided units. Thus, two kinds of control modules, i.e., a palette function (provided by a number equal to that of palettes) of calling and executing each module in the palette and a palette chain function (one in each program) of calling and executing such a
10 palette function itself, have been necessary.

[0058]

After the topological sort, all the programs are integrated and ordered. Thus, only one control module for calling the modules in the programs is necessary for all the
15 programs.

<Chapter 2: Integration of basic structures and topological sort>

A specific procedure of the topological sort will be described by using requirements of the Example 1.

20 1. Adjacency matrix of basic structures

FIG. 4 is a directed graph showing the word relation of FIG. 3. The words a to d which are input words belonging to the basic structure BS1 become endmost start points of a word network represented by the directed graph. Similarly, as shown in the Table 1, it is necessary for the word X_r , which is a routing action element belonging to the basic structure BS1, to have a value (to get result of Boolean operation to decide a next basic structure to be executed) that values of the word a or b and the words c and d. Therefore, the word X_r is linked with the four words by arrows starting separately from the four words. As for the word e belonging to the basic structure BS2 and the word f belonging to the basic structure BS3, since the words c, d and X_r are necessary to establish values of them, they are linked with these three words by arrows starting separately from the three words. As for the output word g belonging to the basic structure BS1, since the word e and the word f are necessary to establish its value, it is linked with these two words by arrows starting separately from the two words.

[0059]

Next, the system of the Example 1 shown by the directed graph of FIG. 4 will be described by using an adjacency matrix. First, an adjacency matrix F1 indicating the basic structure BS1 (901) of the input word is represented by the following equation 1.

Adjacency matrix of basic structure BS1 (input words)

[0060]

[Equation 1]

$$F1 = \begin{matrix} & \begin{matrix} a & b & c & d & x_r \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ x_r \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \end{matrix}$$

As elements of the BS1 (input words) are the words a to d and x_r , the matrix F1 comprises these elements. The words a to d that are input words have no start-point words.

5 Accordingly, rows of end-point words a to d (intersection points with columns of start-point words a to d and x_r) become "0" indicating nonuse as start-point words. The word x_r that is a routing action element has the words a to d as the start points as shown in the directed graph of FIG. 4. Accordingly,
10 in a row of the end-point word x_r , intersection points with these words become "1" indicating that they are start-points.

[0061]

Next, an adjacency matrix F2 indicating a combination of the basic structures BS2 and BS3 (hereinafter, basic
15 structures BS2 & BS3) is represented by the following equation 2.

Adjacency matrix of basic structures BS2 & BS3

[0062]

[Equation 2]

$$F2 = \begin{matrix} & e & f \\ \begin{matrix} e \\ f \end{matrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{matrix}$$

As elements of the BS2 & BS3 are only words e and f, the matrix F2 comprises these two elements. The words e and f are independent of each other (i.e., neither of the words uses another as a start-point word), and thus all of intersection points thereof become "0".

[0063]

An adjacency matrix F3 of a set of output words of the basic structure BS1 is represented by the following equation 3

10 Adjacency matrix of basic structure BS1 (output word)

[0064]

[Equation 3]

$$F3 = \begin{matrix} & g \\ g & [0] \end{matrix}$$

An element of the BS3 is only a word g. Accordingly, the matrix F3 comprises only one word g1. Since the word g is not used as a start point itself, an intersection point becomes "0".

[0065]

Based on the basic structure units constituting the system of the Example 1, the relations among the elements in the same basic structure have been represented by the adjacency matrixes.

2. Integration of basic structures by connection adjacency matrix

Next, to integrate the above into one adjacency matrix as the structures of the entire system, description will be made for a connection adjacency matrix that links the basic structures by showing a relation between words in the
5 different basic structures.

[0066]

A relation between a word of a basic structure and a word of other basic structure is represented by a relation of end-point and start-point words as shown in FIG. 4. It can be
10 said that the basic structures are linked together by a relation words belonging to the structures. The followings are obvious from FIG. 4.

1) When considering words of the basic structure BS1 (input) as end-point words, words of all the other basic
15 structures appear after the word of the BS1 (input). Thus, these latter words cannot be start-point words of the former.

2) When considering words of the basic structures BS1 and BS2 as end-point words, the word of the BS1 (input) that appears before the words of the BS2 & BS3 can be start-point
20 words, while an word of the BS1 (output) cannot be a start-point word.

3) When considering the word of the basic structure BS1 (output) as an end-point word, words of the BS2 & BS3 and the BS1 (input) can be start-point words as they appear before the
25 word of the BS1 (output).

[0067]

As a result of the foregoing consideration, it is understood important for the connection adjacency matrix linking the basic structures together that it should indicate

a relation between a basic structure and other basic structure including words that can be start-point words of the words belonging to the former basic structure. It is because as shown in the directed graph, relations between words are all directed from start-point words to end-point words. Thus, relations between basic structures are all directed from a basic structure including start-point words to a basic structure including end-point words, while there is no reverse relation (in the connection adjacency matrix indicting such a relation among basic structures, intersection points all become 0).

[0068]

Hereinafter, the connection adjacency matrixes that link the basic structures (indicating the relation from the basic structure including the start-point word to the basic structure including the end-point word) of the Example 1 will be considered one by one.

[0069]

An equation 4 represents a connection adjacency matrix FC1 which indicates a relation from the basic structure BS1 (input) (including start-point word) to the basic structures BS2 and BS3 (including end-point words).

[0070]

[Equation 4]

$$FCI = \begin{matrix} & \begin{matrix} a & b & c & d & x_r \end{matrix} \\ \begin{matrix} e \\ f \end{matrix} & \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{matrix}$$

The adjacency matrix FC1 indicates that, when the words e and f as elements of the basic structures BS2 & BS3 are end-point words, the words a to d and x_r as elements of the BS1 (input) are start-point words for them, showing a relation
 5 between the two basic structures. As shown in FIG. 4, the words e and f use the words c, d and x_r as start-point words. Thus, in the FC1, intersection points of the rows of the end-point words e and f with the start-point words c, d and x_r become "1", and intersection points with the other start-point
 10 words become "0".

[0071]

An equation 5 represents an adjacency matrix FC2 which shows a relation from the basic structure BS1 (input) and the basic structures BS2 & BS3 (start-point words) to the basic
 15 structure BS 1 (output) (end-point word) .

[0072]

[Equation 5]

$$FC2 = g \begin{matrix} a & b & c & d & e & f \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{matrix}$$

The adjacency matrix FC2 indicates that, when the word g
 20 as an element of the basic structure BS1 (output) is an end-point word, the words a to d, x_r , e, and f as elements of the BS1 (input), BS2, and BS3 which appear before them in the directed graph are start-point words, showing a relation between the two groups. As shown in FIG. 4, the word g uses
 25 the words e and f as start-point words. Thus, in the FC2, intersection points of the row of the end-point word g with the start-point words e and f become "1", and intersection

points with the other start-point words become "0".

[0073]

The connection adjacency matrixes having meanings of linkage among the basic structures constituting the system of the Example 1 have been described. To simplify explanation of a process of integrating the matrixes into one adjacency matrix for the entire system, connection adjacency matrixes among the basic structures which have no relations will be described below. An equation 6 represents a connection adjacency matrix FC3 showing that there is no relation from the basic structures BS2 & BS3 (start-point words) to an the basic structure BS 1 (input) (end-point word).

[0074]

[Equation 6]

$$FC3 = \begin{matrix} & e & f \\ \begin{matrix} a \\ b \\ c \\ d \\ x_1 \end{matrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{matrix}$$

15

An equation 7 represents a connection adjacency matrix FC4 indicating that there is no relation from the basic

structure BS1 (output) (start-point word) to the basic
structure BS1(input) (end-point word).

[0075]

[Equation 7]

$$FC_4 = \begin{matrix} & g \\ a & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ b & \\ c & \\ d & \\ x_r & \end{matrix}$$

5

An equation 8 represents a connection adjacency matrix
FC5 indicating that there is no relation from the basic
structure BS1 an (output) (start-point word)to the basic
structures BS2 & BS3 (end-point words).

10

[0076]

[Equation 8]

$$FC5 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

An equation 9 represents the system of the Example 1 as one adjacency matrix F by using the adjacency matrixes F1 to F3 and the connection adjacency matrixes FC1 to FC5 described above.

[0077]

[Equation 9]

$$F = \begin{bmatrix} F1 & 0 & 0 \\ FC1 & F2 & 0 \\ FC2 & & F3 \end{bmatrix} = \begin{matrix} & a & b & c & d & x_r & e & f & g \\ \begin{matrix} a \\ b \\ c \\ d \\ x_r \\ e \\ f \\ g \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{matrix}$$

In the above equation, a matrix constituted of F1, F2, F3 and FC1, FC2, and 0 (means an adjacency matrix whose intersection points are all 0 on a right side of "F =", which correspond to FC3, FC4, and FC5) indicates how an adjacency

matrix and a connection adjacency matrix of basic structure units are integrated to represent the entire system. Next, changing these elements into words, an adjacency matrix having a, b, c, d, x, e, f, and g set as elements of the matrix are
5 shown on the right side.. FIG. 5 more easily shows how F1, F2, F3, FC1, FC2, FC3, FC4, and FC5 are integrated to constitute F. In FIG. 5, a portion 1001 is F1, a portion 1002 is F2, a portion 1003 is F3, a portion 1004 is FC1, and a portion 1005 is FC2. In a connection adjacency matrix whose intersection
10 points are all 0, FC3 is a portion 1006, FC4 is a portion 1007, and FC5 is a portion 1008.

[0078]

Thus, one system that comprises two or more basic structures can be represented by one adjacency matrix.

15 [0079]

As the entire system can be represented by one adjacency matrix as described above, topological sort disclosed in the Patent Document 6 by the same applicant is applied to the adjacency matrix of the system to rearrange the programs of
20 the unit words in an optimal order so that value states of all the words can be established by a minimum number of execution times (i.e., generation of all output words is completed).

[0080]

In the adjacency matrix F of the system of the Example 1,
25 as shown in FIG. 6, intersection points with start-point words having values of 1 are all within a left lower triangle (1301). As described above in the Chapter 1, it means that the words are arranged in an optimal order. Thus, it can be said that the adjacency matrix F of the equation 9 has been subjected to

topological sort, and a word order has been set to enable completion of generation of output words by a minimum number of execution times if an initial value is provided to a state vector of a word.

5 [0081]

Now, effects of executing the topological sort will be complemented. According to the program of word units arranged in an optimal execution order by the topological sort, one-round processing can generate values of all output words.

10 However, in the case of the entire system, to be more accurate, realization by a "minimum number of execution times" is a proper expression. It is because in the entire system, a process of repeatedly using the same calculation equation may be included, where a temporary recording area are provide as
15 in the case of totaling processing. Needless to say, "repetition" in this case is not useless iteration. A number obtained by subtracting 1 from the total number of data becomes the minimum number of iteration times. Using a connection adjacency matrix, such a totaling process can be
20 represented by one adjacency matrix.

[0082]

As one example of the search technique, the topological sort has been described. However, other search techniques may be used. The adjacency matrix definition has been included to
25 facilitate understanding of the invention. However, it is not an essential step. In other words, search such as topological sort may be reached directly from definition of an inter-declaration relation or via creation of a directed graph without defining an adjacency matrix. This is also within the

teachings of the present invention, and capable of achieving the object of the invention.

3. Verification of adjacency matrix of program

5 In this section, as the adjacency matrix F of the system described in the above section functions as a function of generating values of the output words, and has been subjected to the topological sort, the completion of the generation of all the output words by the minimum number of execution times will be verified by a calculation operation of multiplication
10 with word state vectors disclosed in the Patent Document 6 by the same applicant. It can be said that the adjacency matrix F functions as a function after the value of the output word g is decided. Additionally, a system of Example 5 does not include iteration processing similar to the aforementioned
15 totaling processing. It can therefore be said that processing is completed by a minimum number of execution times if a state of a value of an output word g is established by one-round processing.

[0083]

20 In the system of the Example 1, states of value of words before the user executes inputting, i.e., an initial value of a state vector X of words is represented by the following equation 10 while all the elements are "null (undecided)" as disclosed in the Patent Document 6 by the same applicant.

25 [0084]

[Equation 10]

$$X = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{bmatrix} \begin{bmatrix} \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \end{bmatrix}$$

Calculation processing of multiplying the adjacency matrix F (i.e., system of the Example 1) of the equation 9 by the state vector X of the equation 10 is executed (i.e., an input is made to the system of the Example 1). A state of a value of a word changed by FX calculation processing will be described for each of end-point words.

(1) Calculation of the end-point word a during one-round processing

Calculation of the end-point word a during one-round processing is as follows.

$$a = 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null}$$

$$= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1)$$

$$= (+1)$$

As it has no start-point word, a state of a value of the input word a is established even if it is multiplied by a state of any value, and changed from null to (+1). In other words, a value is established by an input. As a result, a state of a value of each word after completion of the word a line during one-round processing is as shown in Table 2 below.

[0085]

[Table 2]

	a	b	c	d	x_r	e	f	g
State of value	+1	null	null	null	null	null	null	null

(2) Calculation of the end-point word b during one-round processing

Calculation of the end-point word b during one-round processing is as follows because the state of value of the words of the Table 2 is used.

$$b = 0 \cdot (+1) + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null}$$

$$= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1)$$

$$= (+1)$$

As it has no start-point word, a state of value of the input word b is established even if it is multiplied by a state of any value, and changed from null to (+1). As a result, a state of a value of each word after completion of the word b line during one-round processing is as shown in Table 3 below.

[0086]

[Table 3]

	a	b	c	d	x_r	e	f	g
State of value	+1	+1	null	null	null	null	null	null

In the system of the Example 1, the words a and b are

buttons chosen alternatively on the screen. Accordingly, states of value of both words are established by selecting one of the buttons.

(3) Calculation of the end-point word c during one-round processing

Calculation of the end-point word c during one-round processing is as follows because the state of value of the words of the Table 3 is used.

$$\begin{aligned} c &= 0 \cdot (+1) + 0 \cdot (+1) + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + \\ &0 \cdot \text{null} \\ &= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) \\ &= (+1) \end{aligned}$$

As it has no start-point word, a state of value of the input word c is established even if it is multiplied by a state of any value, and changed from null to (+1). As a result, a state of value of each word after completion of the word c line during one-round processing is as shown in Table 4 below.

[0087]

[Table 4]

	a	b	c	d	x _r	e	f	g
State of value	+1	+1	+1	null	null	null	null	null

(4) Calculation of the end-point word d during one-round processing

Calculation of the end-point word d during one-round processing is as follows because the state of value of the words of the Table 4 is used.

$$\begin{aligned} d &= 0 \cdot (+1) + 0 \cdot (+1) + 0 \cdot (+1) + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + \\ &0 \cdot \text{null} \\ &= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) \end{aligned}$$

= (+1)

As it has no start-point word, a state of a value of the input word d is established even if it is multiplied by a state of any value, and changed from null to (+1). As a result, a state of value of each word after completion of the word d line during one-round processing is as shown in Table 5 below.

[0088]

[Table 5]

	a	b	c	d	x_r	e	f	g
State of value	+1	+1	+1	+1	null	null	null	null

(5) Calculation of the end-point word x_r during one-round processing

Calculation of an end-point word x_r during one-round processing is as follows because the state of value of the words of the Table 5 is used.

$$\begin{aligned} x_r &= 1 \cdot (+1) + 1 \cdot (+1) + 1 \cdot (+1) + 1 \cdot (+1) + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} \\ &= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) \\ &= (+1) \end{aligned}$$

As the states of value of the words a to d which become start-point words of the word x_r have been established, i.e., definition equation execution conditions (1) and (2) of the word x_r which is a routing action element shown in the Table 1 have been satisfied, a state of a value of the word x_r is changed from null to established (+1). As a result, a state of value of each word after completion of the word x_r line during one-round processing is as shown in Table 6 below.

[0089]

[Table 6]

	a	b	c	d	x _r	E	f	g
State of value	+1	+1	+1	+1	+1	null	null	null

(6) Calculation of the end-point word e during one-round processing

Calculation of the end-point word e during one-round processing is as follows because the state of value of the words of the Table 6 is used.

$$e = 0 \cdot (+1) + 0 \cdot (+1) + 1 \cdot (+1) + 1 \cdot (+1) + 1 \cdot (+1) + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null}$$

$$= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1)$$

$$= (+1)$$

As the states of values of the words c, d and x_r which become start-point words of the word e have been established, i.e., values of the start-point words of a definition equation c + d, and the routing action element x_r has specified a basic structure to be executed, a state of value of the word e is changed from null to established (+1). As a result, a state of value of each word after completion of the word e line during one-round processing is as shown in Table 7 below.

[0090]

[Table 7]

	a	b	c	d	x _r	e	f	g
State of value	+1	+1	+1	+1	+1	+1	null	null

(7) Calculation of the end-point word f during one-round processing

Calculation of the end-point word f during one-round processing is as follows because the state of value of the words of the Table 7 is used.

$$f = 0 \cdot (+1) + 0 \cdot (+1) + 1 \cdot (+1) + 1 \cdot (+1) + 1 \cdot (+1) + 0 \cdot (+1) + 0 \cdot \text{null} + 0 \cdot \text{null}$$

$$\begin{aligned} &= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) \\ &= (+1) \end{aligned}$$

As the states of value of the words c, d and x_r which become the start-point words of the word f have been established, a state of value of the word f is changed from null to (+1). As a result, a state of value of each word after completion of the word f line during one-round processing is as shown in Table 8 below.

[0091]

10 [Table 8]

	a	b	c	d	x _r	E	f	g
State of value	+1	+1	+1	+1	+1	+1	+1	null

In the system of the Example 1, execution conditions of the words e and f are alternative conditions. Thus, when one of the conditions is established to be true, the other condition is established to be false, thereby establishing values of both. For example, when the BS2 is selected, "definition equation execution condition of the word e = true" is decided, and a value is provided to the word e, thereby establishing a state of value. On the other hand, in the case of the word f, "definition equation execution condition = false" is decided, and a value is not provided to the word f, thereby establishing a state of value. Thus, when the routing action element selects one of the basic structures, states of value of the words e and f have been established.

(8) Calculation of the end-point word g during one-round processing

Calculation of the end-point word g during one-round processing is as follows because the state of value of the word of the Table 8 is used.

$$\begin{aligned} g &= 0 \cdot (+1) + 0 \cdot (+1) + 0 \cdot (+1) + 0 \cdot (+1) + 0 \cdot (+1) + 1 \cdot (+1) + 1 \cdot (+1) + \\ &0 \cdot \text{null} \\ &= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) \\ &= (+1) \end{aligned}$$

5 As the states of value of the words e and f which become
the start-point words of the word g have been established, a
state of value of the word g is changed from null to (+1). As
a result, a state of value of each word after completion of
the word g line during one-round processing is as shown in
10 Table 9 below.

[0092]

[Table 9]

	a	b	c	d	x_r	E	f	g
State of value	+1	+1	+1	+1	+1	+1	+1	+1

Thus, one round of calculation processing of all the
word lines of the adjacency matrix F has been finished.
15 Accordingly, the Table 9 shows a result of the FX. A state
vector X_1 of the following equation 11 is a result of the FX.

[0093]

[Equation 11]

$$X_1 = \begin{bmatrix} a & +1 \\ b & +1 \\ c & +1 \\ d & +1 \\ x_r & +1 \\ e & +1 \\ f & +1 \\ g & +1 \end{bmatrix}$$

As described above, in the adjacency matrix F, the word unit programs are arranged in an optimal order by the topological sort to be applied. Thus, states of value of all the words are decided by one-round calculation processing, thereby deciding a value of the output word g.

4. Removal of routing action element

Execution conditions of the routing action element are equal to those for executing the definition equation of a set (one or more) of all words belonging to a basic structure specified by the routing action element. It is because all words of the basic structure are executed when the conditions of the routing action element are satisfied. Accordingly, the execution conditions of the route operation element can be changed to those of the words belonging to the basic structure specified by the routing action element.

[0094]

Specific description in the case of the system of the Example 1 is as follows. First, the definition equation and the definition equation execution conditions of the words x_r that is routing action element, e and f are shown in Table 10 below.

[0095]

[Table 10]

Word	Definition equation	Definition equation execution conditions
x_r	BS2 is selected	(1) $a = \text{true}$ (button a is pressed, and a has a value), $c = \text{true}$ and $d = \text{true}$
	BS3 is selected	(2) $b = \text{true}$ (button b is pressed, and b has a value), $c = \text{true}$ and $d = \text{true}$
e	$c + d$	x_r selects BS2
f	$d \times d$	x_r selects BS3

Thus, the x_r condition (1) " $a = \text{true}$, $c = \text{true}$ and $d = \text{true}$ " is replaced by the execution condition " x_r selects BS2" of the word e belonging to the basic structure BS2 selected when this condition (1) is established. The condition (2) " $b = \text{true}$, $c = \text{true}$ and $d = \text{true}$ " is replaced by the execution condition " x_r selects BS3" of the word f belonging to the basic structure BS3 selected when this condition (2) is established. As a result, the definition equation and the definition equation execution conditions become as shown in Table 11 below.

[0096]

[Table 11]

Word	Definition equation	Definition equation execution conditions
e	$c + d$	$a = \text{true}$ (button a is pressed, and a has a value), $c = \text{true}$ and $d = \text{true}$
f	$c \times d$	$b = \text{true}$ (button b is pressed, and b has a value), $c = \text{true}$ and $d = \text{true}$

As shown in Table 11, when the execution conditions of the routing action elements are replaced by the execution

conditions of the words of the selected basic structure, linkage among the BS1, BS2 and BS3 is established by the execution conditions of the words of the BS2 and the BS3.

Thus, the routing action elements only indicate that

5 conditions for executing a next basic structure have been satisfied (input words c and d have values, and button a or b has been pressed). This means that no change occurs in the meaning of the system even if the routing action elements are removed. Accordingly, after replacing the execution
10 conditions of the routing action elements by the execution conditions of the words of the basic structure selected, the routing action elements can be removed from the adjacency matrix of the entire system.

[0097]

15 FIG. 7 is a directed graph showing a relation among the words of the system of the Example 1 after removal of the routing action elements. In FIG. 7, the word x_r that is a routing action element is removed, and a relation between the words e and f and start-point words is shown in accordance
20 with the definition equation and the definition equation execution conditions defined in the Table 11. That is, the word e has the word a (definition equation execution condition "execution button a is pressed"), and the words c and d (definition equation "c + d" and definition equation execution
25 condition "c = true and d = true") as start-point words. The word f has the word b (definition equation execution condition "execution button b is pressed") and words c and d (definition equation "c + d" and definition equation execution condition "c = true and d = true") as start-point words. A relation

from the words e and f to the word g is similar to that shown in FIG. 4.

[0098]

The system of the Example 1 after removal of the routing operation elements is represented as follows by adjacency matrixes in accordance with the directed graph of FIG. 7. F1' of an equation 12 is an adjacency matrix of the basic structure BS1 (input). The word x_r that is a routing action element is removed from the elements.

10 [0099]

[Equation 12]

$$F1' = \begin{matrix} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}$$

F2' of an equation 13 is an adjacency matrix of the basic structures BS2 & BS3.

15 [0100]

[Equation 13]

$$F2' = \begin{matrix} e & f \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{matrix}$$

F3' of an equation 14 is an adjacency matrix of the basic structure BS1 (output).

[0101]

5 [Equation 14]

$$F3' = g[0]$$

FC1' of an equation 15 is a connection adjacency matrix indicating a relation from the basic structure BS1 (input) to the basic structures BS2 & BS3. As the routing action element
10 has been removed, the word X_r as an element of a start-point word is removed. The words e and f respectively have new words a and b as start-point words because of changing of the execution conditions of the routing action element as shown in the Table 11. Thus, for the end-point word e, the number of
15 intersection points with the start-point word a is 1 in addition to those with the start-point words c and d. For the end-point word f, the number of intersection points with the word b is 1 in addition to those with the start-point words c

and d.

[0102]

[Equation 15]

$$FCI' = \begin{matrix} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} e \\ f \end{matrix} & \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \end{matrix}$$

5 FC2' of an equation 16 is a connection adjacency matrix indicating a relation from the basic structures BS2 & BS3 to the basic structure BS1 (output). As the routing action element has been removed, the word X_r as an element of a start-point word is removed.

10 [0103]

[Equation 16]

$$FC2' = \begin{matrix} & \begin{matrix} a & b & c & d & e & f \end{matrix} \\ g & \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{matrix}$$

15 FC3' of an equation 17 is a connection adjacency matrix indicating that there is no relation from the basic structures BS2 & BS3 to the basic structure BS1 (input). As the routing action element has been removed, the word X_r as an element of a start-point word is removed.

[0104]

[Equation 17]

$$FC3' = \begin{matrix} & e & f \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{matrix}$$

FC4' of an equation 18 is a connection adjacency matrix indicating that there is no relation from the basic structure BS1 (output) to the basic structure BS1 (input). As the routing action element has been removed, the word X_r as an element of a start-point word is removed.

[0105]

[Equation 18]

$$FC4' = \begin{matrix} & g \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{matrix}$$

indicating that there is no relation from the basic structure BS1 (output) to the basic structures BS2 & BS3.

[0106]

[Equation 19]

$$FC5' = \begin{matrix} & g \\ e & \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ f & \end{matrix}$$

5

The adjacency matrix F of the system can be represented by an adjacency matrix F' of the following equation 20 when the conditions of the routing action elements are replaced and the routing action elements are removed.

10

[0107]

[Equation 20]

$$F' = \begin{bmatrix} F1' & 0 & 0 \\ FC1' & F2' & 0 \\ FC2' & & F3' \end{bmatrix} = \begin{matrix} & a & b & c & d & e & f & g \\ a & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ b & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ c & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ d & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ e & \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \\ f & \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \\ g & \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{matrix}$$

In the above equation, a matrix constituted of $F1'$, $F2'$, $F3'$ and $FC1'$, $FC2'$, and 0 (means adjacency matrixes having 0

at all intersection points, which corresponds to FC3', FC4', and FC5') indicates how an adjacency matrix and a connection adjacency matrix of basic structure units are integrated to represent the entire system. Next, on the right side, changing these elements into words, we have an adjacency matrix having a, b, c, d, e, f, and g as elements of the matrix. FIG. 8 more easily shows how F1', F2', F3', FC1', FC2', FC3', FC4', and FC5' are integrated to constitute F'. In FIG. 8, a portion 1201 is F1', a portion 1202 is F2', a portion 1203 is F3', a portion 1204 is FC1', and a portion 1205 is FC2'. In a connection adjacency matrix whose intersection points are all 0, FC3' is a portion 1206, FC4' is a portion 1207 and FC5' is a portion 1208.

[0108]

Thus, one system that comprises two or more basic structures can be represented as one adjacency matrix by removing the routing action elements.

[0109]

Next, topological sort is applied to the adjacency matrix of the system to rearrange the programs of the unit words in an optimal order so that value states of all the words can be established by a minimum number of execution times (i.e., generation of all output words is completed). In the adjacency matrix F', as shown in FIG. 9, intersection points with start-point words having values of 1 are all within a left lower triangle (1601). As disclosed in the Patent Document 5 by the applicant, it means that the words are arranged in an optimal order. Thus, it can be said that the adjacency matrix F' of the equation 20 has been subjected

to topological sort, and a word order has been set to enable completion of generation of output words by a minimum number of execution times if initial values of a state vector of words is provided.

5 5. Verification of structure when the routing action element has been removed

10 In this section, it is verified that the adjacency matrix F' described in the above section, which indicates the structure of the system when the routing action elements has been removed, has the same functional function as that of the adjacency matrix F of the structure before the removal of the routing action element. Hereinafter, a process of obtaining a value of the output word g by providing a word state vector to the adjacency matrix F' will be described. If values of word state vectors X_m have all been established as a result of m times calculation process, F' can be said to be a function which functions similarly to F . Additionally, it will be verified that the execution of the topological sort will enable completion of generation of output words by the minimum number of execution times. It can be said that the adjacency matrix F' of the system of the Example 1 can be executed by a minimum number of execution times if a state of a value of an output word g is established by one-round process.

[0110]

25 As described above, for the initial values of the word state vectors, all the elements are "null (undecided)". Thus, a state vector X' of words to be first provided to F' is represented by the following equation 21.

[0111]

[Equation 21]

$$X' = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{bmatrix} \begin{bmatrix} \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \\ \text{null} \end{bmatrix}$$

Calculation processing of multiplying the adjacency matrix F' (i.e., system of the Example 1) of the equation 20 by the state vector X' of the equation 21 is executed (i.e., an input is made to the system of the Example 1). A state of value of words changed by $F'X'$ calculation processing will be described one by one of each end-point word.

(1) Calculation of the end-point word a during one-round processing of $F'1$

Calculation of the end-point word a during one-round processing is as follows because a state of value of the X' is used.

$$\begin{aligned} a &= 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} \\ &= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) \\ &= (+1) \end{aligned}$$

As it has no start-point word, a state of value of the input word a is established, and changed from null to (+1).

In other words, a value is established by an input. As a result, a state of value of each word after completion of the word a line during one-round processing is as shown in Table 12 below.

5 [0112]

[Table 12]

	a	b	c	d	e	f	g
State of value	+1	null	null	null	null	null	null

(2) Calculation of the end-point word b during one-round processing of F'1

10 Calculation of the end-point word b during one-round processing is as follows because the state of value of the words of the Table 12 is used.

$$\begin{aligned} b &= 0 \cdot (+1) + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} \\ &= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) \\ &= (+1) \end{aligned}$$

15 As it has no start-point word, a state of value of the input word b is established, and changed from null to (+1). As a result, a state of value of each word after completion of the word b line during one-round processing is as shown in Table 13 below.

20 [0113]

[Table 13]

	a	b	c	d	e	f	g
State of value	+1	+1	null	null	null	null	null

(3) Calculation of the end-point word c during one-round processing of F'1

25 Calculation of the end-point word c during one-round processing is as follows because the state of value of the words of the Table 13 is used.

$$\begin{aligned}c &= 0 \cdot (+1) + 0 \cdot (+1) + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} \\&= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) \\&= (+1)\end{aligned}$$

As it has no start-point word, a state of value of the
5 input word c is established, and changed from null to (+1).
As a result, a state of value of each word after completion of
the word c line during one-round processing is as shown in
Table 14 below.

[0114]

10 [Table 14]

	a	b	c	d	e	f	g
State of value	+1	+1	+1	null	null	null	null

(4) Calculation of the end-point word d during one-round
processing of F'1

Calculation of the end-point word d during one-round
processing is as follows because the state of the value of the
15 word of the Table 14 is used.

$$\begin{aligned}d &= 0 \cdot (+1) + 0 \cdot (+1) + 0 \cdot (+1) + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} \\&= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) \\&= (+1)\end{aligned}$$

As it has no start-point word, a state of value of the
20 input word d is established, and changed from null to (+1).
As a result, a state of value of each word after completion of
the word d line during one-round processing is as shown in
Table 15 below.

[0115]

25 [Table 15]

	a	b	c	d	e	f	g
State of value	+1	+1	+1	+1	null	null	null

(5) Calculation of the end-point word e during one-round

processing of F'1

Calculation of the end-point word e during one-round processing is as follows because the state of value of the words of the Table 15 is used.

5
$$\begin{aligned} e &= 1 \cdot (+1) + 0 \cdot (+1) + 1 \cdot (+1) + 1 \cdot (+1) + 0 \cdot \text{null} + 0 \cdot \text{null} + 0 \cdot \text{null} \\ &= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) \\ &= (+1) \end{aligned}$$

As the states of the values of the words a, c and d which become start-point words of the word e have been
10 established, a state of value of the word e is changed from null to (+1) to be established. As a result, a state of value of each word after completion of the word e line during one-round processing is as shown in Table 16 below.

[0116]

15 [Table 16]

	a	b	c	d	e	f	g
State of value	+1	+1	+1	+1	+1	null	null

(6) Calculation of the end-point word f during one-round processing of F'1

Calculation of the end-point word f during one-round processing is as follows because the state of value of the
20 words of the Table 16 is used.

$$\begin{aligned} f &= 0 \cdot (+1) + 1 \cdot (+1) + 1 \cdot (+1) + 1 \cdot (+1) + 0 \cdot (+1) + 0 \cdot \text{null} + 0 \cdot \text{null} \\ &= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1) \\ &= (+1) \end{aligned}$$

As the states of value of the words b, c, and d which
25 become start-point words of the word f have been established, a state of value of the word f is changed from null to (+1) to be established. As a result, a state of value of each word after completion of the word f line during one-round

processing is as shown in Table 17 below.

[0117]

[Table 17]

	a	b	c	d	e	f	g
State of value	+1	+1	+1	+1	+1	+1	null

(7) Calculation of the end-point word g during one-round

5 processing of F'1

Calculation of the end-point word g during one-round processing is as follows because the state of value of the words of the Table 17 is used.

$$g = 0 \cdot (+1) + 0 \cdot (+1) + 0 \cdot (+1) + 0 \cdot (+1) + 1 \cdot (+1) + 1 \cdot (+1) + 0 \cdot \text{null}$$

10 $= (+1) + (+1) + (+1) + (+1) + (+1) + (+1) + (+1)$

$$= (+1)$$

As the states of value of the words e and f which become start-point words of the word g have been established, a state of value of the word g is changed from null to (+1) to be established. As a result, a state of value of each word after completion of the word g line during one-round processing is as shown in Table 18 below.

15

[0118]

[Table 18]

	a	b	c	d	e	f	g
State of value	+1	+1	+1	+1	+1	+1	+1

Thus, one round of calculation processing of all the word lines of the adjacency matrix F' has been finished.

Accordingly, the state of the value of the word of the Table 18 becomes a result of the F'X'. A state vector X'_1 of the following equation 22 is a result of the F'X'.

20

[0119]

25

[Equation 22]

$$X'_1 = \begin{bmatrix} a & +1 \\ b & +1 \\ c & +1 \\ d & +1 \\ e & +1 \\ f & +1 \\ g & +1 \end{bmatrix}$$

As described above, in the adjacency matrix F' , the word unit programs are arranged in an optimal order by the topological sort. Thus, values of all the words are established by one-round calculation processing, thereby establishing a value of the output word g .

[0120]

It can be understood that the adjacency matrix F' , where the routing action element has been removed, by providing an initial value (inputting is executed) to a state vector, can establish the states of value of all the words to generate the value of the output word g as in the case of the adjacency matrix F . Thus, it has been verified that the adjacency matrix F which is the system having the routing action element and the adjacency matrix F' which is the system having no routing action element have the same processing function as system.

[0121]

As described above, when an entire system includes plural basic structures, by integrating them and by applying search (e.g., topological sort) to the integrated structure we can provide much better performance in application of autonomous development technology of Lyee (registered trademark) software or the like. However, for example, when the plural basic structures are not integrated and stay devided, performance as equal as that of the present invention cannot be obtained. If order is not optimized by the search technique such as topological sort, performance as equal as that of the present invention cannot be obtained.

6. Reuse of program

Processing that needs reuse of the program is repeated even if topological sort is executed.

All kinds of programs (algorithms) can be described only by a recursive method. All kinds of programs written in procedural type programming language (e.g., C, java or the like) can be written by a while program which plays the same role as that of the recursive program. This means that the while program can do anything theoretically.

However, a program method based on words cannot use itself as a start-point word. For example, $a = a + b$ cannot be used for the program based on words. It is because a value of a is not changed once decided. Accordingly, recursive use needs an area to temporarily hold the value of a and a method of clearing it. The held value is used as a start point. For this purpose, an area for temporarily holding contents must be provided. This area is called a temporary word. A structure action element (vector of clear) is defined for clearing.

According to the Lyee methodology, an action element is a kind of word having a predicate-structure. A structure of a program code is similar to that of the word. A main difference from the word is that two or more words are simultaneously treated. The meanings of them are an interface to the outside, a routing for control structures or a clearing areas.

[0122]

Adjacency matrixes of programs (matrixes of all programs) are integrated into one matrix by using divided adjacency matrixes and connection matrixes for changing states only. The connection matrix indicates that words of the other group are used as start-point words. Accordingly, a mathematical model of the program based on words can be used. Thus, all the usual algorithms expressed in procedural type language can be represented by the program structure based on words.

[0123]

Next, integration of adjacency matrixes by a connection matrix will be described by using (Example 2) as an example of totaling processing which needs a temporary word such as the a = a + b.

(Example 2) Values of records x_2 , x_6 of certain words of a file are totaled sequentially from a head, and a totaling result is put to words x_3 , x_7 .

[0124]

According to the Lyee methodology, as a value cannot be overwritten once decided, description is in a form of $x_3 = x_1 + x_2$, and this is reused. x_1 is a word to substitute for a value

of x_3 . The value of x_3 is transferred to a temporary area x_4 to substitute for x_1 at the time of next reuse. This is called a "temporary word". FIG. 10 is a directed graph showing the Example 2. The words x_1 , x_2 , and x_3 are cleared after x_3 is transferred to x_4 . Words x_5 , x_6 and x_7 are respectively reuses of the words x_1 , x_2 , and x_3 .

[0125]

An adjacency matrix (one path matrix) of a part of a program reused in FIG. 10 is F1 below.

10 [0126]

[Equation 23]

$$F1 = \begin{matrix} & x_{1,5} & x_{2,6} & x_{3,7} \\ \begin{matrix} x_{1,5} \\ x_{2,6} \\ x_{3,7} \end{matrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} & : \text{one path matrix} \end{matrix}$$

To reuse F1, connection matrixes to integrate F1 and reused F1 are F2 and F3. These have functions of transmitting states. The matrixes show each end-point word use which word as tart-point words.

15

[0127]

[Equation 24]

$$F2 = \begin{matrix} & x_1 & x_2 & x_3 \\ x_4 & \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \end{matrix} : \text{connection matrix from } x_1, x_2, x_3 \text{ to } x_4$$

$$F3 = \begin{matrix} & x_4 \\ x_5 & \begin{bmatrix} 1 \end{bmatrix} \\ x_6 & \begin{bmatrix} 0 \end{bmatrix} \\ x_7 & \begin{bmatrix} 0 \end{bmatrix} \end{matrix} : \text{connection matrix from } x_4 \text{ to } x_5, x_6, x_7$$

F2 is a connection matrix from x_1 , x_2 and x_3 to x_4 . F3 is a connection matrix from x_4 to x_5 , x_6 , and x_7 . F2 and F3 are understood respectively as action elements for an output from x_3 to x_4 and an input from x_4 to x_5 . An adjacency matrix of the entire program for reusing F1 is F.

[0128]

[Equation 25]

$$F = \begin{matrix} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{matrix} : \text{entire system matrix}$$

10 The adjacency matrix of the entire program is represented by using a one-round matrix and a connection matrix. Accordingly, it is possible to use the mathematical

model of the program based on words. Thus, all the usual algorithms expressed in procedural type programming language can be realized by programming based on words and by its structure.

5

<Chapter 3: Specific example using the present invention>

<Mathematical algorithm of topological sort>

Before describing a specific method, we describe one algorithm for applying topological sort to a sequence of elements in a partial order.

10

Lyee specifications are represented as follows in a word system.

$$x = F(x) \quad (\text{Equation 2-1-3})$$

Here, functional F is defined as follows.

15

$$F(x)(i) = f_i(x(1), x(2), \dots, x(n)), \quad i = 1, 2, \dots, n$$

(Equation 2-1-2)

B is called a chain when a subset B , that is not an empty set, of an ordered set (A, \leq) is an all-order set under \leq , and called a reverse chain when no binaries of B can be compared. As the number of end points equal to or more than the number of start points defined on the right side is defined by one-round of iteration processing, F is monotonic. Thus, iteration lines generated one after another repeating F is called a chain and represented the following equation.

20

25

[0129]

[Equation 26]

$$(F^k)_{k \in N}$$

Here, k denotes the number of iteration times. The minimum number of iteration times is a maximum chain size. Generally, a partial order set can be divided into reverse chains by the following theorem.

[Theorem]

When n is a maximum chain size of the partial order set (A, \leq) , A is divided into n reverse chains which are elements of each other.

This theorem is proved by mathematical induction. The following can be said when the theorem is applied to all words of the partial order set. That is, all the words are divided into n word sets among which there is no order. A length (size) n of an iteration column (equation 26) denotes a largest among chains obtained from a relation between start and end points of words, i.e., the minimum number of iteration times necessary until synchronization.

The proof of the theorem provides one algorithm which applies topological sort to a sequence of elements of the partial order set when the sequence is provided. That is, A_1, A_2 are sequentially obtained from an input side until $A_{n+1} = 0$ is reached in which A_n is a reverse chain, elements of A_n are arranged in a proper order, then elements of A_{n-1} are arranged in a proper order, and elements of A_1 are lastly arranged in a proper order, whereby topological sort can be applied in an

ascending order. Here, A_n is a set of the largest elements. Here, in other words, it is a set of words closest to an output side. If the words thus arranged are executed from A_1 (input side), all the words are established one after another
5 by one-round processing.

1. Method integrating divided structures

Basic structures are integrated into one, and words are rearranged. A program generated by this method has the following structure.

- 10 1) A W04 logical element for generating a value is set on W03.
- 2) The W03 and W04 logical elements are integrated to constitute a word. Equivalent words are integrated into one word.
- 3) Words of all basic structures are set on the W4 of a start
15 screen. At this time, execution conditions of routing action elements are spread on the words. The routing action elements are not spread.
- 4) For all the words and the input/output operation elements, a relation between a start point and a basic element is
20 represented by a directed graph, and topological sort is applied by the aforementioned method to decide an order and to arrange them. Iteration in a palette is not set because it is not necessary. Clearing of a palette area is not set because it is not necessary.
- 25 5) A flag of a control area and flags of the input/output operation elements need to be cleared, and thus are left as they are.

As a result,

- 1) the words and the input/output operation elements belonging

to the same basic structure have the same execution conditions because the execution conditions of the routing action elements of the basic structure are equally spread, and thus a group is created after the sort.

- 5 2) Consequently, basic elements are regrouped by the route execution conditions, and the basic elements of the group are arranged in an execution order.

2. Specification of an example

10 Requirements of this example concern a case of a student grade management system. According to the system, a person of general secretaries registers a student name, and teachers register results of tests, whereby total evaluation can be registered at the end of term. This system comprises three screens of an "initial screen" shown in FIG. 11, a "student
15 registration screen" shown in FIG. 12, and a "result administration screen" shown in FIG. 13, and four files of an "ID & password file" registering ID's and passwords of managers and teachers, a "student name file" registering student names and ID, a "test result file" registering results
20 of tests for each student, and a "total score file" registering term-end total score for each student.

25 The initial screen is a screen displayed when the result administration system is started. The system is finished by pressing an Exit button. When inputs are made to a field of an ID and a password, and an OK button is pressed, the system refers (reads) to input data and data registered in the ID & password file, and determines whether the input data is a general secretary person's or a teacher's, or belong to neither. If a result of the determination shows that the ID

and the password is a manager's, the student registration screen is displayed, the result administration screen is displayed if it is a teacher's, and the initial screen is displayed if they belong to neither.

5 [0130]

10 The student registration screen is a screen on which the person of the general secretaries register student names and student ID numbers. It is a screen displayed when the ID and the password determined on the initial screen is a general secretary person's. During displaying, the system refers to the student name file, and displays data of a registered student in a field of a list of students. After the displaying of the screen, a student name and a student ID number are input to a field of new registration, and input data are registered (written) in the student name file by pressing a registration button. Pressing of a return button enables return to the initial screen.

 [0131]

20 The result administration screen is a screen for registering and referring to results of tests and total score of each student. It is displayed when the ID and the password determined on the initial screen is a teacher's. During screen displayed, the system refers to the student name file, and prepares lists of student names and student ID numbers of the accessing teacher's as selection candidate lists to be displayed in text boxes of student names. When the person of the general secretary presses a right-end button of the text box to display the student name list, and selects (inputs) a student name and an ID number to be referred to, the system

refers to the test result file to calculate and display test results of the input student and an average point of the test results. If total score has been registered, the system simultaneously refers to the total score file to display the total evaluation. To register a new test result, data of a selected student is input to the test result registration field, and the registration button is pressed. The system registers the new data in the test grade file, simultaneously updates and displays contents of the test result field to contents containing the new data, and recalculates and displays an average point. To register total score, total score of the selected student is input to a total score registration field, and the registration button is pressed. The system registers data of the total score in a total evaluation file, and simultaneously displays the data registered in the total evaluation field. The system can return to the initial screen by pressing the return button.

FIG. 14 is a process route diagram showing requirements of the result administration program, and in this case the program is divided into conventional basic structures.

Partial iteration of the same processing to display a list of test results for one student, or the like, and totaling calculation for obtaining an average point are included, and the aforementioned basic processing operations are all included.

3. Implementing procedure of case

A language to be used is C++, a function of C++ is used only for a screen part, and the rest is described only by a C grammar. A database management system is Access 97 with DOA.

At the time of designing, requirements divided into nine basic structures were implemented in an integrated form of one basic structure. For program generation, by using LyeeAll2 (automatic code generation tool of Lyee structure program), its template (form of basic element and control program) was changed to automatically generate codes of a basic structure integrated type Lyee program. FIG. 15 shows a part of the new integrated type program code. A 7-th line of the new program indicates route execution conditions, i.e., execution conditions in process definition, and basic elements are accordingly grouped again. The basic elements in the W02 palette are arranged in an order after sort. For words, generation equations alone remain indicated in 11 to 14-th lines.

Comparison of the Lyee structure program generated in the divided basic structure with the new program by the Lyee methodology is as follows. The number of steps of the execution and time are in the case of a program of a requirement "one is selected from 7 students, and results of 5 tests are displayed". A CPU processing time does not include access time of a database, and a total process time is a total time including the access time of the database. A processing time was obtained as a difference between a start and an end by adding a timestamp function to the program.

4. Result and evaluation

1) Result

[0132]

[Table 19]

	Conventional Lyee structure program	Invention applied Lyee program	
--	--	-----------------------------------	--

Number of program lines	213,559lines	6,889lines	31%
CPU process time	150msec	25msec	15%
Total process time	470msec	345msec	73%

A reduction in execution codes is caused by a small word structure and one basic structure. Main changes are as follows.

1. A predicate-structure comprises about 21 lines. This usually becomes a generation equation alone which comprises one substitution equation line.
2. W03 generation conditions and a W04 generation equation are integrated into one.
3. For action elements, routing action elements constitute one IF sentence. Input/output operation elements remain almost as they are.
4. 80% of clearing action elements is removed.
5. One basic structure needs about 24 KB excluding words and operation elements. In this case, 9 basic structures are integrated into one.

[0133]

Reductions in number of execution steps and execution time are mainly caused by a reduction in iteration. Main changes are as follows.

1. All the palettes are iterated at least twice, and the basic structures are iterated at least three times. These numbers of times become one.
2. The program code is reduced by 31%.
3. However, the total process time is not reduced so much. It is because the access time of the database is predominant.
4. In real-time processing that does not use database

processing, for example, in process control of a building monitoring system, a CPU time is predominant. Accordingly, the new method is very advantageous in size and processing time of the program in the case of process control.

5 2) Evaluation

As compared with the program based on the conventional structure type programming, the following can be said. At present, problems remain in the program by the conventional structure type programming based on the specification, and the
10 conventional method is limited to qualitative comparison because it depends on programmer's skills.

1) Designing of program

Execution conditions of words can be understood as action execution conditions of process definition, whereby
15 designing difficulties of the conventional Lyee methodology are reduced. Understanding of specification in a declarative form which is an advantage of the Lyee methodology is realized by process definition and word definition in the new program. Thus, in the new program, as a control procedure is realized
20 by the process definition, program designing is not necessary. According to the conventional method, certain designing is necessary based on specifications.

2) Size of program

As compared with the program by the conventional
25 structure type programming, in the new program, there are no codes apparently increased in structure.

3) Execution speed

According to the new program, the number of iteration times is limited to a necessary minimum. Thus, there is no

reason for a lower execution speed as compared with the conventional program.

<Chapter 4: Conclusion>

5 In the present circumstances, according to the program generated by a tool based on the Lyee methodology, specifications divided at the time of designing are developed into program modules in the divided state. Use of information regarding an order explicitly expressed in the requirements is
10 surely effective for improving designing accuracy and efficiency. However, according to the program of the Lyee structure, problems that can be statically solved before execution are dynamically solved during the execution. Thus, a code size is large, and a processing speed is low. There is
15 accordingly room for improvement. A program can be generated without changing its meaning and without dividing basic structures from the start while utilizing the effects of efficient designing. Accordingly, all the words are integrated into one area. Thus, when a correct order is
20 decided among all the words by topological sort in a directed graph of the word relation, iteration for securing ordinality is made unnecessary (see Patent Document 6).

 Thus, designing a process route diagram, i.e., dividing requirements, is very useful for accurately understanding the
25 system requirements. The designing the route process diagram means that words having common elements in output conditions are regarded as logical record units. Allocation of the common elements of the conditions to the routing action elements is a great advantage for efficiently defining user

requirements.

[0134]

Additionally, at the time of generating a program code and executing the program, integration of systems divided into
5 basic structures is useful. It is because word unit programs of the entire system can be arranged in an execution order so that it can complete generation of output words by a minimum number of execution times.

[0135]

10 As described above, in the system that comprises declaration execution modules of word units, processing of the declaration execution modules for generating output data is completed by eliminating useless iteration and by a minimum number of execution times, and the task of reducing a program
15 size is solved by the following means.

1) The entire system is shown in a process route diagram by setting a set (logical body) of words having the same output conditions as one basic structure. Accordingly, it is possible to accurately understand execution conditions
20 (conditions for the routing action element to specify the basic structure) common to the declaration execution belonging to the same basic structure.

2) The routing action element is a target of topological sort, and requirements thereof are defined by a word relation as in
25 the case of logical elements defining a word relation. That is, definition is made as to which basic structure to be executed is specified (definition equation) under established conditions (execution conditions of definition equation).

3) The routing action element is removed by adding the

definition equation execution conditions of the routing action
element to the execution conditions of the modules belonging
to the basic structure specified by the routing action element
when the conditions are established. By setting the execution
5 conditions of the routing action element on the execution
conditions of the modules of the specified basic structure,
linkage among the basic structures (i.e., linkage among the
words) is established even after the routing action element is
removed. Thus, it is possible to integrate all the programs
10 into one.

The declaration execution modules are rearranged in an
optimal order by applying topological sort for the declaration
execution modules (not including the routing action element)
of all the programs thus integrated into one. Accordingly, it
15 is possible to execute the program by avoiding useless
iteration and by a minimum number of execution times.

[0136]

The present invention is not limited to the
aforementioned embodiment and examples. Various changes can
20 be made within the scope of technical ideas of the invention.
For example, the invention can be realized as a business
method, software development device, a software development
support device, software for realizing such function by a
computer, or a recording medium/dedicated machine having the
25 software mounted therein. Moreover, as described above,
needless to say, the present invention can be realized by a
method, software having its function, a device/tool (including
software itself) having the software mounted therein, or a
system.

[0137]

For example, FIG. 16 is a functional block diagram showing a configuration disposed when the present invention is implemented as one of a program (software) for generating "development target software", a program generator, a program processor, a tool (including both device and software), a software development device, a software development support device, and a software development management device according to a different embodiment of the invention.

[0138]

The overall control section 1601 has a function of executing overall operation control, timing control, input/output control or the like of the program (software), the program generator, the program processor, the tool (including both device and software), the software development device, the software support device, or the software development management device, and it is realized as a dedicated chip/circuit having the function, software (including software as a tool) for causing a computer to execute the function, a recording medium recording the software, or a processor/management device/tool having the recording medium mounted thereon.

[0139]

FIG. 17 is a flowchart showing an operation of the present invention implemented as one of the program (software), the program generator, the program processor, the tool (including both device and software), the software development device, the software support device, and the software development management device having the aforementioned

configuration.

[0140]

Explanation for the both figures is omitted.

[0141]

5 Thus, according to the present invention having the
aforementioned configuration, in the program that comprises
the declaration execution modules of word units, processing of
the declaration execution modules for generating output data
can be completed by avoiding useless iteration and by a
10 minimum number of execution times, and the program size can be
reduced.

[0142]

 According to the different embodiment of the present
invention, each of a program (software) for generating
15 "development target software", a program generator, a program
processor, a tool (including both a device and software), a
software development device, a software development support
device, and a software development management device can be
configured by comprising: means for defining, based on user
20 requirements implemented as one program, all necessary
declaration execution units to satisfy the requirements that
are L2 processing (attribute check processing of input word),
L processing (value generation processing of output word), I2
processing (logical record input processing), and O4
25 processing (logical record output processing), which are
decided from declarations by the word being composed of a
word name, a definition equation, execution conditions of the
definition equation, input/output attribute, and attributes of
word value and by the logical record of the word with access

conditions; means for defining a (partial) order relation among all the defined L2 processing, L processing, I2 processing, and O4 processing; means for applying topological sort to the L2 processing, L processing, I2 processing, and O4 processing for which the (partial) order relation is defined in the second step; and means for arranging a predetermined code sequence of the declaration execution unit based on Lyee methodology in accordance with an order of the declaration execution units rearranged in the third step.

10 [0143]

Furthermore, the present invention is realized by the software produced by the aforementioned "method of generating development target software", and a recording medium having the software mounted thereon, or a device (hardware) having the software mounted thereon. In this case, the present invention can be configured as a code sequence based on the Lyee methodology in accordance of declaration units with an order rearranged by topological sort applied based on a (partial) order relation defined from the requirements declared by the word units, where declaration execution units, for user requirements mounted as one program, of all necessary for satisfying the requirements, i.e., L2 processing (attribute check processing of input word), L processing (value generation processing of output word), I2 processing (logical record input processing), and O4 processing (logical record output processing), which are decided from declarations by the word being comprised of a word name, a definition equation, execution conditions of the definition equation, input/output attribute, and attributes of a word value and by

the logical record of the words with access conditions.

[0144]

The present invention is realized as software as a form of a software code used for producing the software by the
5 aforementioned "method of generating development target software", and a recording medium having the software mounted thereon or a device (hardware) having the software mounted thereon.

[0145]

10 Furthermore, the present invention can be realized as an extraction method of information (document (paper, data)) extracted from the requirements by the aforementioned "method of generating development target software), the information (document (paper, data)) extracted by the extraction method, a
15 method of using the extracted information, an information recording medium having such information mounted therein, software having the coded information extraction method/using method, a recording medium/device (hardware) having the software mounted thereon, or information extracted from
20 software development requirements having correlated pieces of information to enable realization of such.

[0146]

According to the invention, many changes and modifications can be made by those who have usual knowledge in
25 the technical field. Thus, the invention should not be limited to the configurations or the operations strictly described with reference to the drawings. Accordingly, proper changes and equivalents can all be within the scope of the invention. The invention has been described in detail by way

of specific embodiment and example. However, many modifications, substitutions, and changes can be made without departing not only from the scope of claims of the invention but also from the scope of the invention defined in all the disclosed items.

[0147]

The application of the invention should not be limited to the foregoing description or explanation, or specific appreciation or combinations of the shown elements. Other embodiments are possible, and the invention can be used and implemented by various methods. Moreover, the terms and the phrases used here are only descriptive but not limitative.

[0148]

Thus, as easily understood by those who have usual knowledge in the technical field, the disclosed basic concept can be easily used as a basis for designing other structures, methods and systems for implementing some objects of the invention. Accordingly, such equivalent appreciation is within the scope of the claims of the invention as long as it does not depart from the gist and the scope of the invention.

[0149]

The method of realizing the idea of the invention has been described in detail in the order of the directed graph representation, the adjacency matrix calculation, the topological sort, and the rearrangement. However, these are not absolutely essential elements. For example, the idea of the invention may be realized directly in an order of adjacency matrix calculation → topological sort → rearrangement without executing directed graph representation.

[0150]

Additionally, rearrangement may be made by combining various search methods such as breadth-first search, iterative deeping, heuristic search, hill climbing, best-first search, Euler's single stroke of a brush, Dijkstra method, and the like with the directed graph.

[0151]

Alternatively, the basic idea of the invention may be realized by combining such various search methods after the adjacency matrix calculation.

[0152]

Needless to say, the technical idea of the invention can be realized and used as e.g., an automatic development device, an automatic development program of computer software, or a recording medium, a transmission medium, or a paper medium recording the automatic development program, and in a category of a computer/device on which the automatic development program is mounted, a client server form for executing the automatic development program, or the like.

[0153]

Furthermore, the present invention is applied not only to a computer system equipped with a single processor, a single hard disk drive, and a single local memory, but also to a system equipped with a plurality of optional or combined processors or memory devices as options of the system. The computer system includes an elaborate calculator, a hand type computer, a laptop/notebook computer, a minicomputer, a main frame computer, a supercomputer, and a processing system network combination thereof. An optional proper processing

system that operates in accordance with the principle of the invention may be used instead, or can be used in combination with the above.

[0154]

5 Needless to say, the technical idea of the invention can deal with all kinds of programming languages. Moreover, the technical idea of the invention can be applied to all kinds/functions of application software.

[0155]

10 According to the invention, various changes, additions, substitutions, expansions, reductions, and the like can be made within similar ideas, equivalents, and the scope of the technical idea. Even when software produced by using the invention is mounted on a secondary product and commercialized,
15 a value of the invention is not reduced.

[Industrial Applicability]

[0156]

 According to the present invention thus defined,
20 preprocessing is automatically carried out to avoid iteration, and an object program is mathematically generated by the Lyee (registered trademark) methodology based on the preprocessed word unit program thus obtained. In other words, it is possible to automate a process from sophistication (alignment)
25 of user requirements to generation of an object program. As a result, the invention provides great effects to a software industry, such as great increases in efficiency, productivity, and quality of software production.

[Brief Description of the Drawings]

[0165]

FIG. 1 are conceptual diagrams explaining a concept of cells according to an embodiment of the present invention.

FIG. 2 is an explanatory diagram of a screen of a system of Example 1 according to the embodiment of the present invention.

FIG. 3 is an explanatory diagram showing words of the system of the Example 1 by basic structure units according to the embodiment of the present invention.

FIG. 4 is a directed graph showing a relation among the words of the system of the Example 1 according to the embodiment of the present invention.

FIG. 5 is a diagram explaining a process of representing the entire system of the Example 1 by one adjacency matrix F according to the embodiment of the present invention.

FIG. 6 is a diagram explaining that the adjacency matrix F of the embodiment of the present invention has been subjected to topological sort.

FIG. 7 is a directed graph showing a relation among the words when a route operation element is removed from a structure of the system of the Example 1 according to the embodiment of the present invention.

FIG. 8 is a diagram explaining a process of representing the entire system by one adjacency matrix F' when the route operation element is removed from the system of the Example 1 according to the embodiment of the present invention.

FIG. 9 is a diagram explaining that the adjacency matrix F' of the embodiment of the present invention has been subjected to topological sort.

FIG. 10 is a directed graph of Example 2 according to the embodiment of the present invention.

FIG. 11 is a diagram showing an "initial screen" of the Example 2 according to the embodiment of the present invention.

5 FIG. 12 is a diagram showing a "student registration screen" of the Example 2 according to the embodiment of the present invention.

10 FIG. 13 is a diagram showing a "grade management screen" of the Example 2 according to the embodiment of the present invention.

FIG. 14 is a process route diagram showing a structure in which requirements of a grade management program of the Example 2 of the embodiment of the present invention and this program are divided into conventional basic structures.

15 FIG. 15 shows a part of an integrated type new program code according to the embodiment of the present invention.

20 FIG. 16 is a functional block diagram showing a configuration of functions provided when the present invention is implemented as one selected from a program (software) for producing "development target software", a program generation device, a program processor, a tool (including both of a device and software), a software development device, a software development support device, and a software development management device according to a different
25 embodiment of the present invention.

FIG. 17 is a flowchart showing an operation of the present invention implemented as one selected from the program (software), the program generation device, the program processor, the tool (including both a device and software),

the software development device, the software development support device, and the software development management device, which has the above configuration according to the embodiment of the present invention.

5 [Description of Symbols]

[0158]

- 1 0 1 Synchrononization
- 1 1 0 1 Basic Structure B S 1 (Input)
- 1 1 0 2 Basic Structure B S 2
- 10 1 1 0 3 Basic Structure B S 3
- 1 1 0 4 Basic Structure B S 1 (Output)
- 1 2 0 1 Adjacency Matrix F 1'
- 1 2 0 2 Adjacency Matrix F 2'
- 1 2 0 3 Adjacency Matrix F 3'
- 15 1 2 0 4 Connection Matrix F C 1'
- 1 2 0 5 Connection Matrix F C 2'